Linear Programming

Eighth in a series of LP surveys highlights recent trends in profession's most popular software.

By Robert Fourer

This is the eighth in a series of surveys of software for linear programming, dating back to 1990. As in the case of earlier surveys, information has been gathered by means of a questionnaire sent to software vendors by the editors of OR/MS Today. Results are summarized by product in the tables following this article, after which contact details for further information are listed by vendor.

Products listed in this survey are concerned, at the least, with minimizing or maximizing linear constraints subject to linear equalities and inequalities in numerical decision variables. All products provide for continuous variables that may take any values between their bounds, and many also accommodate integer variables that are limited to whole-number values in some way. The respectively continuous and discrete problems that use these variables are commonly distinguished as linear programs (LPs) and integer or mixed-integer programs (IPs or MIPs), but for convenience "LP software" is used herein as a general term for the packages covered, and "LP" refers to problems that may or may not have some integer variables.

Some of the listed products handle other kinds of discrete variables and constraints, as well as varied nonlinearities and even problems outside of optimization. Indeed, there has been a trend toward greater generality in recent years. This survey focuses on developments and trends in the linear programming and related integer programming aspects of the software. The ordering of topics is roughly parallel to the organization of the tables, and terms in bold correspond to table headings.

Additional responses are welcome and will be added to the Web version of the survey. To learn more, write to Online Projects Manager Patton McGinley, patton@lionhrtpub.com, or go directly to the survey at www.lionhrtpub.com/ancill/lpsurvey2005.shtml.

Types of Packages

Although the products surveyed have a common purpose and share many aspects of design, they are best understood as incorporating two complementary but fundamentally different types of software. Solver software takes an instance of an LP model as input, applies one or more solution methods, and returns the results. Modeling software does not incorporate solution methods; it is typically designed around a computer modeling language for expressing LP models, and it offers features for reporting, model management and application development, in addition to a translator for the language.

Numerous solver and modeling products have been developed as independent
Applications. Thus, solvers typically support links to many modeling systems, and modeling systems work with many solvers. In some cases the two may be acquired as separate products and linked by the purchaser, but more commonly they are bought in bundles of various kinds. Most modeling system developers arrange to offer a variety of bundled solvers, providing modelers with an easy way to benchmark competing solvers before committing to purchase one. Some solver developers also offer bundles with modeling systems. A number of the latter developers also offer integrated systems that provide a modeling environment specifically for their own solvers. Many variations on these arrangements are possible, so that prospective purchasers are well advised to confirm the details carefully.

**Interfaces to Other Software**

Since optimization models are usually developed in the context of some larger algorithmic scheme or application (or both), the ability of LP software to be embedded is often a key consideration. Thus, although virtually any of the listed products can be run as an independent application in some kind of stand-alone mode, many are available in callable library form, and an increasing number can be accessed as class libraries in an object-oriented framework. Solver systems have long been available in these ways, with an application-specific calling program taking the place of a general-purpose modeling environment. Modeling systems have increasingly also become available for embedding, however, so that the considerable advantages of developing and maintaining a modeling language formulation can be carried over into application software that solves instances of a model. It is possible to embed an entire modeling system, or a particular model or an instance of a model; not all systems provide all possibilities, so some study is necessary to determine which products are right for a given project in this respect.

Most commercial LP software libraries are distributed as binaries for linking into the user's applications. In addition to the few offering source code that are listed in our table, several non-commercial solvers make their code available through one of the standard open-source licenses (http://www.opensource.org/). Currently the most prominent open-source LP codes include GLPK (www.gnu.org/software/glpk/glpk.html) and lp_solve (groups.yahoo.com/group/lp_solve/), which also have interfaces to MathProg, an open-source modeling language; and CBC/CLP, available at the COIN-OR site (http://www.coin-or.org/) along with packages for building specialized branching schemes for solving mixed-integer programs. Open source is ideal in situations where the greatest degree of flexibility is required, such as in creating new algorithms and algorithmic schemes, or in putting together specialized application packages that require optimization problems to be solved somewhere inside. But where the emphasis is on building models, solving instances and analyzing results, it makes more sense to use commercial software that someone else has gone to the trouble of compiling and debugging. Demonstration versions of commercial codes are often available (see further below) and even for open-source projects there are often compiled binaries available for the most common platforms.

The application development environments provided by spreadsheet and database programs have proved to be particularly attractive for embedding of LP software. At the least, most LP modeling environments can read and write common spreadsheet and database file formats. Spreadsheet packages can also accept solver add-ins whose appeal to users and convenience for development are widely appreciated. The solver add-ins that come packaged with spreadsheet products are effective only for small and easy problems; much more powerful LP software add-in options are available for separate purchase. Several developers of scientific and statistical packages also offer LP software add-ins specifically for their products.

Virtually all LP modeling systems and solvers can also handle model instances expressed in simple text formats, especially the "MPS" format dating back many decades and various "LP" formats that resemble textbook examples complete with + and = signs. These formats mainly serve for submitting bug reports and for
communicating benchmark problems. Modeling systems use much more general and efficient formats for communicating problem instances to solvers and for retrieving results. Each uses its own format, unfortunately, so that every modeler-solver link requires a different translation. Possibilities for a superior standard form for model instances continue to receive growing attention, particularly XML-based forms that would facilitate the integration of LP software with Web communication standards. Many interconnected issues must be addressed in devising a new standard, however, and in the case of LP there is the additional complication of wanting a standard that will extend well to other kinds of optimization models of interest.

Platforms
The range of supported platforms has seen little change. Windows remains universal. Linux, Solaris and other varieties of Unix are quite common, especially where no sophisticated graphical user interface is involved. A half dozen products listed as having Apple MacOS versions did not have such versions at the time of the previous survey; most Mac versions are created by porting to the underlying Unix shell of MacOS System X, however, rather than by creating new versions that conform to the standard Macintosh look and feel.

Multiprocessor versions for shared memory have become more widely available, as multiple processors have become more of a standard for high-end workstations. Undoubtedly there will be support as well for the new "dual core" designs that put two processors on one chip. Support for distributed memory remains rare, despite increasing general interest in "grid computing" and networks of workstations. Distributed processing seems a natural fit for integer programming branch-and-bound methods, which solve independent subproblems at nodes of a huge search tree, but promising experiments with this approach do not seem to have led yet to much commercial support.

A very fast dual-processor PC with multiple gigabytes of memory is thus perhaps the platform of choice for very large LPs or very hard MIPs. LP software continues to become increasingly available for the 64-bit processors that are necessary for effective support of more than 4 gigabytes of physical memory and that may offer better performance even for smaller memory amounts.

Sizes
Most LP software can take advantage of whatever computer memory is available. For MIP solvers, the ability to take advantage of available disk space to store part of the search tree is also valuable.

Size-limited demo versions (also often called student versions) permit experimentation with small problem instances. They are typically full-featured versions of the software, limited only by being restricted to problem instances of up to a few hundred variables and constraints. Several modeling systems are conveniently packaged in demo versions with one or more solvers.

Most modeling language and solver developers will arrange to provide full versions of their software for testing for a limited time. A number of developers also make their products conveniently available for testing and comparison over the Internet, via the NEOS Server (www-neos.mcs.anl.gov). NEOS imposes no problem size restrictions and is free of charge. It does not guarantee confidentiality or availability of service, but these are not the key issues in initial stages of testing.

Prices
The table shows individual commercial licenses running from $500 to $5,000 and more, but many vendors quote prices only on request. Special terms are often available for multiple purchases, floating licenses (which permit a certain number of copies to be used anywhere in a larger network) and site licenses. Since solver
performance varies considerably from problem to problem and from product to product, buyers are well advised to benchmark problems of interest before deciding which products to price out.

**Algorithms**

Solution methods have continued to be refined for speed and reliability. For linear programs a choice between primal simplex, dual simplex and interior-point methods is now standard. The bag of tricks that make up the typical MIP branch-and-bound solver continues to grow, even after decades of attention, with increasingly sophisticated features such as branch-and-cut, branch-and-price and feasibility-seeking heuristics becoming available to a broader range of users. These refinements make more integer programs tractable but also place more responsibility on the user to study and select wisely among available options. Although developers choose default settings carefully, often by means of proprietary heuristics, defaults cannot be relied upon to work well for all hard MIPs; even a simple upgrade from one version of a package to the next may necessitate an adjustment to the option settings.

**Problem types**

Developers continue to look for ways to address their customers' needs by supporting new specializations and generalizations of LPs and MIPs.

In the area of discrete optimization, the ideas underlying branch-and-bound search for integer programming are sufficiently powerful to handle broader classes of constraint types. Indeed, MIP solvers have long accommodated variables that take values from an arbitrary list (via special ordered sets of type 1, or SOS1 search rules) and objectives or constraints that incorporate non-convex piecewise-linear terms (via SOS2 rules). Many solvers also have special search rules to help with semi-continuous variables, which must either take a prespecified value (usually zero) or lie in a designated continuous range. Additional kinds of logical constraints, such as if-then and all-different, are gradually becoming more common as specialized search techniques are adapted from related developments in constraint programming software. The distinction between integer programming and constraint programming is thus continuing to fade, though it will not disappear for some time yet.

Spreadsheets are also proving to have a strong influence in discreet optimization. A spreadsheet solver uses expressions in certain cells to specify objective and constraint functions, so users naturally expect to be able to employ whatever expressions the spreadsheet program makes available. Quite a few of these are logical, non-smooth or even discontinuous, however, necessitating extensions to existing optimization methods.

**Convex quadratic** objectives and constraints, in continuous or integer variables, are another popular extension as seen in the table. Convex quadratically constrained quadratic programming further extends to so-called second-order cone programming, and to semidefinite programming where non-negativity of individual variables is generalized to positive semidefiniteness of symmetric matrices of variables. Problems of these types find varied applications in engineering and design, and provide strong approximations to some hard combinatorial problems; some examples can be found at www.gamsworld.org/cone/conelib.htm. Interior-point methods extend to solve these problems, though not with the same ease as LP problems. Modeling languages are only beginning to catch up with these problems; they are likely to become more familiar once modeling systems more fully support them.

A number of products listed in the table can handle some more general nonlinear problems as well. There are quite a few good solvers intended specifically for nonlinear optimization that do not appear, but that can be found in other listings and surveys.
What's next? The complementary slackness conditions for optimality of linear programs can be viewed as a special case of a *complementarity* or *equilibrium* constraint. In the simplest case, such a constraint requires two inequalities to be satisfied, at least one with equality. Specialized solvers have long been available for equilibrium problems consisting entirely of such constraints. But recent work has made great progress in solvers for the more general case of *mathematical programs with complementarity (or equilibrium) constraints* — MPCCs or MPECs for short. These problems have varied applications from economics to engineering, as can be seen in the MPEC Library collected at [www.gamsworld.org/mpec/mpeclib.htm](http://www.gamsworld.org/mpec/mpeclib.htm).

**Robert Fourer** ([www.iems.northwestern.edu/~4er/](http://www.iems.northwestern.edu/~4er/)), professor of Industrial Engineering and Management Sciences at Northwestern University and a member of the Optimization Technology Center of Northwestern and Argonne National Laboratory, is one of the designers of the AMPL modeling language for mathematical programming and the NEOS Server facility for optimization over the Internet. He maintains the Linear Programming and Nonlinear Programming Frequently Asked Questions lists at [www.mcs.anl.gov/home/otc/Guide/faq/](http://www.mcs.anl.gov/home/otc/Guide/faq/).